GLOBAL TERRAIN TEXTURE: LOWERING THE COST

Michael A. Cosman
Principal Engineer
Evans & Sutherland Computer Corporation
Salt Lake City, Utah

Abstract

Texture fulfills two important but different operational requirements. It provides the optical flow necessary to establish visual calibration of position, orientation and velocity. It can also provide realism and geographic correlation when derived from appropriate source material and applied in a geographically specific way to the terrain. The latter approach involves either the use of many individual texture maps that are mosaicked together by the construction of the terrain model, or the use of dedicated texturing hardware that provides a continuous, seamless single texture space. This paper contrasts these two terrain texture approaches, addresses a number of related technical issues, characterizes the strengths and limitations of each approach, discusses the subtleties in their various implementations, and explores the economic and technical issues of source data acquisition and reconciliation. The paper also discusses the migration of high-end IG technology downward into low-cost approaches that can be used within the framework of very modestly priced systems.

Introduction

The global terrain texture requirement was first articulated in a major procurement for the Special Operations Forces Aircrew Training System (SOFATS), and was there addressed by an aggressive all-out high-end system approach (Cosman, 1990). The SOFATS customer wanted photo-specific texture decoration of a half-million square miles of terrain, accomplished in a few days, with no guarantees of appropriate source material for any particular part of the data base. There were some concessions to reality: in deference to the limits of current technology, texture resolution was allowed to vary so as to permit storage on a finite number of very large disks. Most of the data base was to be decorated with quite coarse texture, with a number of higher-resolution corridors and insets.

Presented at the IMAGE VII Conference
Tucson, Arizona 12-17 June 1994

It's fair to say that industry expectations of global terrain texture have mellowed in the intervening years, as source material limitations and economic realities have forced a harder look at what's really needed. The ESIG 4000 system delivered for the SOFATS program can, in fact, digest and display much more texture than its owners can afford to feed it. Acquisition of photo texture at even modest resolutions has proven very costly for large-area data bases. At the same time, other users have obtained excellent operational utility from systems and data bases that make much less extensive use of photo texture. Collectively, we have come to better understand resolution, content, correlation, and realism requirements.

Along the way, increasing use of photographic texture has energized the development of improved tooling. A veritable smorgasbord of native and third-party image acquisition and processing technology is now available. Steadily accumulating operational experience across the whole inventory of simulation systems continues to emphasize the importance of texture resolution, as users trade coarse "real" texels for high-resolution generic texels at an exchange rate that proves photo-realism isn't everything. One consequence of this has been the development of a process for synthesizing photo-realistic texture inside the IG on a just-in-time basis from thematic source data.

The global terrain texture requirement could currently be said to include the following: 1) the terrain texture motif must be *source-correlatable* everywhere, and *photo-recognizable* in specified places; 2) overall, the textural motif must be consistent in content, currency, correlation with other aspects of the simulation, and in more intuitive photographic attributes like color, intensity, and saturation;

3) the resolution of the texture must meet mission optical flow requirements, which can vary throughout the data base depending on the activities to be performed; 4) the implementation of global terrain texture cannot otherwise constrain how the user interacts with the simulated environment--he must retain full freedom to operate in any place, manner or velocity otherwise required of the system; and 5) the motif should contain no distracting artifacts due to the way the texture is acquired, reconciled and mosaicked, attached to the terrain model, or rendered by the IG hardware.

The Importance Of Coherence

All of the processes used to develop, store, apply and render texture make use of certain spatial coherence properties to simplify processing and improve rendered image quality. These properties are naturally preserved internal to the small texture motifs that fit within the historical IG hardware mechanisms originally designed to deal with texture. The global terrain texture motif is, by comparison, relatively infinite, and doesn't fit within those mechanisms. If it is broken into a patchwork of smaller texture motifs that do fit, spatial coherence is interrupted at those patch boundaries. Loss of coherence at patch boundaries generally means that those boundaries will stand out in anomalous ways that thwart the achievement of requirement 5) above. It may also mean the loss of algorithmic and computational shortcuts that have considerable value, and increase the amount of resource required to achieve global terrain texture functionality.

A proper understanding of texture and its coherence issues is essential to understanding why a dedicated hardware global terrain texture mechanism is superior to a patchwork approach. The texture section of the paper discusses these issues in some detail as it lays the groundwork for contrasting these two approaches. The mosaic approach is then characterized, with special attention to complications that arise out of the texture coherence issues. Next, an approach is presented that employs dedicated IG hardware to treat the global terrain texture as a single motif that is not broken or subdivided. Advantages of this mechanism are explained and quantified. Finally, we discuss the source material problem, which is largely economic, and briefly describe an approach that can be used to inexpensively synthesize selected portions of the global terrain texture motif.

Texture

Texture is, in effect, a digital wallpaper that is applied to surfaces in the visual scene, and represents the variation of some attribute on the interior of those surfaces. It is an implicitly regular two dimensional (2D) array of data. Members of the array are called texels (texture cells), and the array is called a texture map. The texture coordinate space is implied by the nature of the array, i.e., its two coordinate axes are defined by the rows and columns of the array. Thus the arrangement and scaling of the array has spatial and visual meaning. A texture map is applied to surfaces in the scene by designating how the texture coordinate space relates to the polygon coordinate space. Most systems allow some real-time modification of this relationship to provide texture motion for special effects.

Texel values can represent changes in any spatially varying surface parameter. Typically texture is used to represent changes in intensity, color, opacity, or thematic content (such as surface material type). Several different textures may be combined on the same surface to achieve new effects, such as independent motions within different layers of smoke, or a transparency overlay whose motif is independent in size and scale from the base color modulation (trees, etc.) Textures of different scales may be combined to increase the distance between repetitions of the texture, and in general to increase the visual variety of texture effects that can be constructed from the comparatively small amount of texture that can be stored within the image generator. As declining hardware costs allow improved texture utilization, an industry baseline for texture seems to be emerging: full-color red/green/blue plus a transparency.

How Texture Is Applied

A great deal of computation must be done within the IG to apply texture. Most of this is pixel (picture-element) rate stuff, so the robustness of texture implementations has been largely determined by advances in hardware technology. The earliest implementations of texture took advantage of the computational simplifications of scanning surfaces in (or parallel to) the coordinate datum, and only

applied texture to the ground plane, or to surfaces parallel to the ground plane. The first widely used texture systems employed a generalized texture transformation to simplify texture scanning on surfaces at arbitrary orientations. One interesting consequence of this approach was that the texture plane didn't need to coincide with the plane of the polygon. Many of the more recent high-end systems do model-space texture scanning, which facilitates the use of non-linear image transforms such as are required for dome projection.

Texture-to-Polygon "Tacking". In general, each polygon carries the information that connects it to its texture. This includes information about what maps are to be used, how they are to be combined and interpreted, and how they connect ("tack") to the polygon. Texture tacking is typically done by specifying texture coordinate pairs (map row/column) for the texels at each polygon vertex. The IG will scan the texture by interpolating these coordinate pairs across the interior of the polygon. Note that the texture is always applied in the plane of the polygon, and that vertex sharing of the texture tack coordinates will provide automatic texture continuity across polygon boundaries--an especially useful property for terrain texture. Also, the tacking paradigm is a particularly intuitive way for humans to understand the geometric association of the texture and polygon spaces.

Boundary Issues. Mathematically, the texture coordinate space is infinite. If a polygon is not contained fully within a single repetition of the texture map, the texture motif is expected to simply repeat enough times to completely fill it. Historical texture development and application processes have anticipated this repetition. If the texture motif is *not* intended to repeat, then the texture map boundary must align with a polygon edge. A single texture map can be applied to a mesh of polygons, but at its boundaries it must coincide with polygon edges. This is a particularly important issue for the terrain texture problem, and will be discussed later.

Texture Aliasing

Aliasing is a general term applied to a wide variety of image quality problems. The term "aliasing" refers to high frequency image content masquerading as low frequency stuff. The general cause of aliasing is insufficient "sampling": the rendering process (particularly at the pixel level) didn't ask enough questions (samples) about what's going on to get a consistent answer about what to draw. The visual effects of aliasing include image crawling, edge stair casing, scintillation, moiré patterns, and general image noise. With regard to texture, aliasing occurs in two (fairly inclusive) domains: when texels (as rendered) are larger than pixels, and when they are smaller. More specifically, texture can alias because the boundaries between texels are not treated correctly, or because texels are skipped over during the sampling and rendering process. Proper control of aliasing depends on exploiting spatial coherence within the texture map.

Texel Edge Aliasing

Texels are discrete values in a 2D array. The array can be thought of as a checkerboard in texture space, where the color or intensity of each square is determined by the texel occupying that square. The scanning process transforms this checkerboard into perspective, applies it to the polygon, and scans it with pixels. During scanning, a view ray associated with each display pixel is extended out into the modeled environment, where it intersects the polygon and a corresponding texel within the texture map. If the scanning process only asks one question ("Which texel square did this pixel hit?"), then the boundaries between texel squares will be rendered as pixel-size jagged "edges" that crawl as the image moves. This effect is most noticeable when texels are larger than pixels (i.e., when you are close to the polygon), but it occurs under all viewing conditions.

Texel edge aliasing can be solved by asking more questions about each pixel--an increase in computational load that is essential for highest image quality. Instead of asking which texel a pixel hit, the IG determines which group of four texels the pixel landed between. It then uses the relative distances from the pixel to each of the four texels to weight their contributions to the pixel. This bi-linear blend of adjacent texels filters the texel boundaries together during rendering, eliminating the jagged edge crawling. Some systems provide selectable degrees of this blending, allowing the modeler to trade off texel edge behavior to achieve a variety of useful effects such as single-texel runway

concrete squares. Bi-linear blend depends on the spatial coherence of neighboring texels--coherence that must be maintained by the processes that create and apply texture. Bi-linear blend has important implications for the IG architecture and the terrain texture problem; both will be discussed later.

Texel Under-Sampling

Texels undergo the same perspective distortions as the polygon, and are generally not square as portrayed on screen. When a polygon is rendered, range, angle or perspective may cause the squares on the texture checkerboard to become smaller than the spacing between pixels. When either dimension of the texel becomes smaller than the pixel, some texels are occasionally skipped over in the rendering process--no pixel sampler hits them. The intermittent inclusion of texels will cause scintillation in the displayed pixels. This can happen even if the perspective texels are larger than a pixel in one dimension--aliasing occurs based on the smaller dimension of the perspective texel. There is a variety of anti-aliasing algorithms for use when texels are smaller than pixels. We will discuss the two most common approaches.

Clamping. More accurately described as *bandwidth limiting in object space*, this was one of the earliest affordable texture anti-aliasing algorithms, and dominates the installed base of commercial flight simulators (Norton, 1982). Clamping treats texture as a two-dimensional bi-polar signal that is attenuated to its mean value as a function of the pixel-to-texel size ratio. The aggressiveness of this attenuation function can be tailored for each polygon and each individual map used on that polygon. This allows the modeler to account for all the factors that contribute to aliasing, like the actual map motif and relative brightness and contrast being portrayed. Clamping allows the most flexible use of limited texture space, and has significantly lower hardware cost than other approaches. It works best in systems that allow several different textures to be combined on each surface, and is a good approach for generic or repeating texture, or texture composed of combined motifs. The ability to combine several textures at different scales (to greatly extend the apparent texture repetition interval) made this the best choice for *generic* terrain texture. Clamp texture is less optimum for non-repeating or photo-themes contained in a single texture layer, such as global terrain texture. Many of the ESIG systems originally delivered with clamp texture are candidates for an upgrade process that substitutes MIP texture and adds the global terrain texture hardware that will be described later in the paper.

Multiple Level-Of-Detail (MIP) Texture. The term "MIP" comes from the Latin "multum in parvo," or "many things in a small place." MIP texture provides a succession of different levels of detail (LOD) for each texture map (Williams, 1983). Each successive LOD is half as many texels in each direction, so each has one-fourth as many total texels as the previous higher level (note: whenever we discuss level of detail, "higher" means the more resolved, complex or defined version). Successive levels are derived by filtering the previous level to remove information that cannot be properly represented by the new, coarser set of texels. The succession of levels continues to a single value that represents the average effect of all the texels in the map. During rendering, the appropriate texture level of detail is selected based on the relationship of smallest texel dimension to pixel size. MIP texture works by ensuring that, at the displayed texture level of detail, every texel gets hit at least once by a pixel sampler. Note that MIP texture uses the spatial coherence of the texture motif to create a new kind of coherence in the level-of-detail domain.

A MIP map consists of its highest level of detail, plus the succession of lower LODs, so it contains 33% more texels than just the highest level (1 + 1/4 + 1/16 + 1/64... = 1.333...) This suggests that the optimum shape for a MIP map is square, and the optimum size (number of texels along each edge) is a power of 2. Rectangular MIP maps can be used with some storage inefficiency, and the lowest (coarsest) levels of detail cannot be filtered optimally since there may not be enough texels in the short dimension. Address computation within the IG is more complex if maps are rectangular, and systems that permit rectangular maps typically limit the dimensions to powers of 2. One important advantage of MIP texture is that the derivation of lower texture levels of detail is completely under user control. Filtering and MIP structure development are done at model time, so the user can select filter algorithms that provide the best performance for his particular application. While MIP texture can provide the mathematically "correct"

merging of detail in the lower LODs, there are times when something different may be better.

MIP maps can have a big impact on IG system architecture, particularly if image quality issues are important. In order to provide continuous transitions between texture LODs, the system must solve the texture look-up and bi-linear blend for the two adjacent LODs that "bracket" the desired continuous LOD (based on pixel/texel size ratio), and then blend these two values to account for being partially "between" them. This all has to happen slightly differently for each pixel, and if this must be done without impacting IG capacity, then the system must have the proper memory architecture and bandwidth to access eight texel values simultaneously--the four within each MIP level required for bi-linear blend, for both levels. It must also provide parallel computation elements to perform the tri-linear blend of these 8 values simultaneously. The lookups and blends can be done sequentially (as in some workstations) but with significant loss of performance. Some workstations that allow selection of the texture anti-aliasing treatment really bog down when asked to do the full MIP tri-linear mode.

Texture Anti-Aliasing Effects. Anti-aliasing algorithms all try to limit the information content of the image to what can be properly presented within the resolution of the display environment. This is always much less than the resolution of the real world, and users are always tempted to tweak the rendering processes to try to let more detail through, even at the expense of more aliasing noise. Texture has some interesting characteristics that exacerbate this issue. As viewed, most texture has been perspectively compressed in one dimension more than the other. Since texture anti-aliasing algorithms must operate based on the most compressed dimension, they inevitably over-compensate and result in loss of texture detail in the less-compressed dimension. At extremely shallow viewing angles, this unequal treatment results in significant loss of texture detail that may be mission critical. Under these conditions, the rate of change of texture LOD may also cause noticeable LOD transition bands. In general, you never get as much resolution out of texture as you think you should, so users must be careful to provide sufficient three-dimensional feature detail in the data base to ensure that the scene is usable under all viewing conditions.

MIP Texture System Issues

If we carefully examine the process of deriving successive texture LODs, we discover some subtleties that have important system-level implications. Each LOD is created by filtering the previous level to half as many texels in each direction. This is done with a filtering operation--a process that considers and weights several neighboring input texels to generate each output texel. Since the filter is several texels wide (in both dimensions), a special problem could arise at the map boundaries if the filter tries to include high LOD texels that are beyond the boundary. If we intend the map to repeat, then we can let this happen, but we will want to wrap the texture address to grab the texel at the opposite edge of the map. If we do not intend for the map to repeat, then we must either know about texels in all the surrounding maps (this assumes some subsequent mosaicking process), or we must re-think the filtering process so it doesn't require texels that are outside the space of the highest-LOD map.

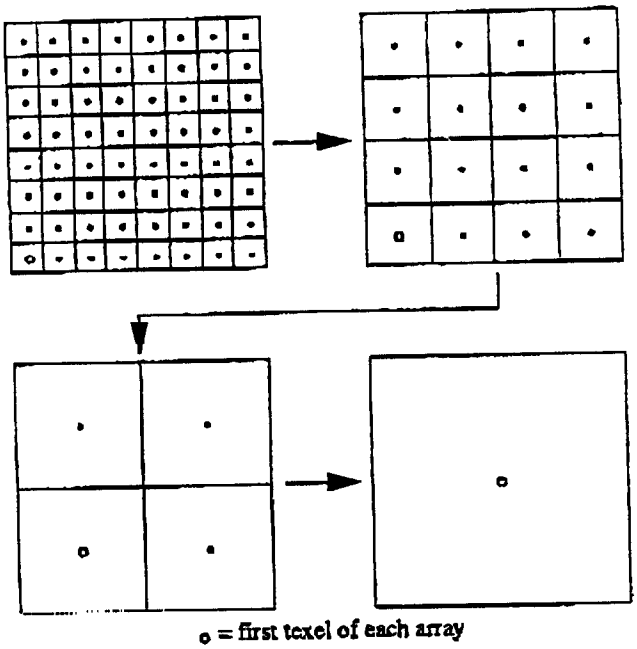

Fig. 1 MIP filtering and coordinate shift.

Such a process might be as simple and intuitive as averaging four high-LOD texels to get each low LOD texel, as Fig. 1 illustrates. As each group of four texels is filtered into a new lower-LOD texel, the *position* of that texel is not coincident with any of the original texels, but has shifted half a texel to the middle of the original group. As the filtering proceeds to

lower and lower LODs, there is a progressive shift of the first texel position towards the middle of the original map space. The final single texel that represents the entire effect of all the original texels exists, mathematically, at the center of the original map--the result of the accumulation of half-texel shifts between each LOD. This shift must be accounted for in the IG hardware if the texture motif is to remain spatially stable and properly correlated with its underlying surface. Putting the half-texel-per-level correction in the IG allows us to defer the question "Does this map repeat or mosaic?" until run time, and greatly simplifies the development of MIP LODs. Both repeating and non-repeating maps are then treated properly in the IG.

Special Issues With Terrain Texture. Generally, global terrain texture doesn't repeat--every texel is geographically specific and gets drawn in just one place in the data base. Conceptually, at least, the texture space must be considered as infinite, continuous and non-repeating. If the data base has any significant geographic extent, then the total quantity of texture will become huge, even for quite large texel sizes. A number of resource limitations throughout the texture development, storage, transmission and rendering stages dictate that the terrain texture motif consist of relatively coarse texels--typically feet or meters large. Even so, special methods must be developed to optimize use of the limited texture storage within the IG hardware, and to read successive geographically related "pages" of texture into the IG as the eye point travels around. At even modest eye point velocities, the quantity of texture being paged (obsoleted and replaced) in the texture memories requires special high-bandwidth data channels. Even so, the system must exploit the geographic coherence of the texture motif to make paging tractable.

New texture generation and application processes must be developed to deal with boundary matching issues, and to allow creation of the texture in its native, single-space context. For affordable texel sizes, the terrain texture alone will provide insufficient visual cueing and optical flow to support critical low-level operations like take-off, landing and hover. Additional textural detail must be supplied by a finer, generic, repeating texture motif superimposed on the terrain texture. The system must have extreme computational resolution to guarantee sub-pixel stability of the texture motif over the large dynamic ranges required, and to guarantee that the motif is continuous across terrain polygon boundaries. Final performance of the system, both visually and dynamically, will depend critically on the overall system-level terrain texture strategy.

Some Simplifying Aspects Of Terrain Texture. While parts of the problem get much harder, there are some saving graces. Both the texture space and the terrain polygon space are visually continuous and non-self-overlapping (with some concessions to the more spectacular cliffs and canyons around...), a particularly beneficial kind of coherence. Because the texture tack space is geographic, there is an intuitive association of the terrain system to the texture system, and a natural way to compute tack coordinates. Terrain environments have extreme area coherence: if you know where you are, you also know exactly what parts of the terrain (and texture) are nearby and likely to need rendering. This can be exploited to help manage the texture paging problem and to smooth out bandwidth requirements. Since you are always above ground, you can exploit things you know about how the texture should behave at its various levels of detail. For example, lower LODs of the terrain texture generally mean you are viewing those surfaces at shallow angles, which might influence your choice of texture filtering algorithms for the lower LODs. Lastly, since terrain topography and feature decoration are typically derived from digital sources that are thematic in nature (DFAD, etc.,) the terrain problem invites additional texture creation and reconciliation strategies based on this data.

Two Terrain Texturing Strategies

We will contrast the use of mosaicked feature texture maps with a dedicated hardware terrain texturing mechanism. The discussion assumes that the application of texture to the terrain must be visually seamless and geographically accurate throughout the data base and through the full spectrum of terrain and texture LODs, and that the texture must be well behaved with regard to aliasing and image quality issues.

The Mosaic Approach

Definition and Characteristics. Terrain is decorated with the usual feature texturing mechanisms, i.e., with (comparatively) small

texture maps, polygon by polygon. A contiguous mesh of texture maps is applied to the terrain model, and each terrain polygon references the map containing its texture, and tells how its vertices and edges tack into the texture map space. Polygon-texture references are embedded in the terrain model in an implicit marriage of the two spaces that occurs piecewise at model time. Seamlessness is achieved by ensuring that all the different map texture spaces are aligned, coincident and related by powers of 2, and that maps begin on, end on and are aligned with polygon edges.

Texture Map Boundary Problems. At the edges of each texture map, the bi-linear blend process will want texels that are mathematically in the neighboring texture map--a consequence of the interruption of spatial coherence in the original motif. There are several ways to deal with this. The IG could saturate the texture address (effectively disabling bi-linear blend) to remain inside the current map--this will cause map boundaries to be noticeable. Alternatively, one could make the map one row and column of texels larger than the polygon to provide the needed blend texels. These texels are duplicated in the neighboring map, and imply some storage overhead. The extra row and column of texels need to be in every texture LOD, and the scaling of the map to the terrain polygon must be adjusted slightly differently at each texture LOD. Since this must be done at the pixel rate, it requires special hardware in the IG. The texture development processes need to account for this as well. If generic, repeating "grit" texture is used, additional care is required to ensure that there are no texture discontinuities in it. Finally, employment of texture insets of differing resolutions (high-resolution insets) is especially tricky.

Scale Interdependencies. While a terrain texture map can cover many terrain polygons, it must end exactly on a terrain polygon edge. If a terrain region is a collection of polygons that are all decorated with the same texture map, then the polygonal boundaries of this region must align with the texture map boundaries. Terrain regions must be rectangular or square along this outside perimeter, even though they can be irregular inside. Terrain polygons cannot be larger than the texture map size, nor can they straddle texture map boundaries. For example, a 1024 by 1024 map of 1 meter texels implies that the largest terrain polygon can only be one kilometer on an edge, and then only if it starts and stops exactly on the implicit grid defined by the texture map mosaic. The benefit of irregular terrain skinning schemes is thus significantly reduced. The complex interdependence of terrain stylization and texture decoration complicates the use of irregular or non-homogeneous terrain skinning, and means that subsequent changes to either strategy may require large-scale re-processing of the entire terrain data base. Local area updates must occur in quanta of the texture map grid spacing.

LOD Interdependencies. Terrain level-of-detail and texture level-of-detail are driven by very different mechanisms that cannot be made to synchronize. Terrain will transition to different LODs based typically on range or roughness; MIP texture will transition based on the relationship of pixel size to perspective texel size. This relationship includes both range effects and orientation effects. Terrain that is at its most complex LOD may require decoration with texture that is at its simplest LOD, if the viewing angle is very shallow. A problem arises when the terrain detail internal to a terrain region has transitioned to its simplest level of detail. Any further simplification must include terrain outside the region covered by the current texture map. The next LOD transition must substitute a new hierarchy of terrain LOD within a new (larger) terrain region, decorated with a new MIP texture hierarchy. At the transition, two versions of the texture must be resident in the IG simultaneously. During the transition, texture that was saturated to its simplest LOD may suddenly (and noticeably) skip several LODs as the proper new versions become available.

Data Base Development. If the feature texturing mechanisms are to be used for global terrain texture, the data base developers must perform a thorough top-level design analysis and carefully define how the system resources will be deployed. They must first determine how much of the available feature texture will be allocated to the terrain problem, and how much will be left for regular feature texture. Data base developers must define the terrain texturing scheme after considering total texel requirements, IG storage constraints, texture paging rates, texture and terrain facet resolutions, and the terrain subdivision strategy. A terrain skinning strategy must be developed that meets the regularity

requirements of the texturing scheme while also staying within the capacities of the IG.

It is likely that each new data base will require the development of a custom process to enforce these complex constraints, with attendant implications on the data base development tooling. Developers will need to construct validation data bases to verify performance constraints. If irregular terrain skinning is used, it might be necessary to complete development and debug of the entire terrain polygon model before finalizing the design strategy for the terrain texture and beginning development of the texture data base.

Modeling tools and processes must be designed to marry texture to terrain on a region basis, while dealing with texture map edge issues and devoting special attention to edge blend and scaling requirements. Texture paging could be controlled by either the real-time system or via information embedded within the data base hierarchy, and tools may need to optimize organization and disk storage of the texture data base to achieve acceptable paging rates. These needs may constrain ability to develop portions of the data base independently and in parallel, lengthening development schedules.

IG Performance Issues. In general, MIP texture maps are stored in some definite scheme within the IG texture memories in order to achieve the texture look-up bandwidth required. A feature texture MIP map must generally be paged in its entirety into contiguous texture memory whenever any portion of it is needed, at any texture LOD. This means that all the texture LODs are required for any terrain that is within the visibility range, not just the texture LODs appropriate for the actual viewing ranges. This has a huge impact on texture paging rates and required texture storage, and affects both final texel resolution and maximum eye point velocity. If the system falls behind in paging texture, performance degradation may not be graceful. Portions of the terrain may be displayed without texture, or with the wrong texture, suddenly changing when the right texture finally arrives in the IG.

Under these conditions the required texture paging rate can be quickly determined from just a few of the design parameters. As the eye point moves, new portions of the data base come within the visibility range as old areas disappear. The amount of area being exchanged is a function of the visibility range and the eye point velocity. The amount of texture within this area is determined by the texture resolution (at the highest LOD). For example, using one meter texels in a system that has a 50 kilometer visibility range, flying at 360 knots (typical parameters for a mission rehearsal application), the system would have to continually stream 25 million texels per second into the IG memories. (185 meters/second X 100,000 meters X 1.333 MIP texels/texel = 25,000,000 texels/second). Note that if all texture LODs for this terrain must be in the IG simultaneously, about 13 gigatexels of hardware texture storage is required.

The Bottom Line. If global terrain texture is implemented with the standard feature texture mechanisms, it's hard to get right, inflexible when done, has recurring costs in tooling and data base top level design, and increases overall data base development costs significantly. Visual and statistical performance is compromised, and in the end it is unlikely that all the troublesome visual artifacts will be successfully suppressed or avoided.

A Low-Cost Dedicated Hardware Approach

The global terrain texture mechanisms originally developed for the ESIG 4000 have been repackaged in a form that provides identical functionality at greatly reduced cost. This option integrates with the ESIG 3000 product and extends the benefits of global terrain texture to a new class of users. The achievement was made possible by incorporating important lessons learned in the SOFATS experience with rapidly advancing memory and ASIC technology. In brief, the approach treats the global terrain texture as a single map that can be arbitrarily large. A suite of mechanisms is developed to deal with this large map, and the catalog of problems outlined in the mosaic approach never occurs because the coherence of the motif is never broken or interrupted. The following section describes these mechanisms.

System Characteristics. Global terrain texture is supported with a complement of dedicated system resources that include modeling tools, real-time software, microcode, texture memory, and rendering hardware. The system provides full color terrain texture

everywhere, and the rendering processes employ sufficient computational precision to maintain texture stability over the global ranges involved. No texture tacking information is required in the terrain polygons themselves--the computation of terrain texture coordinates is performed in the hardware for terrain vertices as needed, at whatever terrain level of detail is instantaneously required. Auxiliary high-frequency textural "grit" can be selected polygon by polygon based on local terrain geology and tailored to the local operational requirements. The system strategy is geared to the highest-resolution texture required, and regions that are textured to lower resolution are rendered with the available texels in a continuous seamless fashion. This provides a very effective mechanism for supporting a data base that consists largely of coarse texels, with occasional high-resolution inset patches.

Texture Memory Organization. Memory is organized and managed so as to keep only the relevant sub portion of this extensive single texture motif resident in the IG. Dedicated texture memory is allocated into independent texture LODs that can be paged and managed separately. The size and organization of the texture memory exploit beneficial characteristics of the terrain texturing problem that were discussed earlier. Since the displayed texture LOD is determined by a combination of range and surface orientation, one can easily determine for any particular range from the eye point, the smallest texels that can be rendered without aliasing (assume, for now, that the surface is looked at straight-on, rather than at a shallow grazing angle). There is no need to carry higher resolution texture than this for any particular range. Thus the smallest size of texels that can safely be used increases with increasing range, and suggests that a fairly modest total amount of texture can suffice to decorate all the terrain from the eye point to the visibility range. The shallow view angles used for most of the terrain mean that the system will actually not even require this resolution, but if the lower MIP LODs are present the required size texels are always available.

Equal amounts of texture memory are allocated to each MIP level of detail for the first number of MIP levels beginning with the highest, or most detailed. Each successive texture MIP level is comprised of progressively larger texels, so it covers four times the geographic area of the next higher level, at half the linear texture resolution (see Fig. 2). As the MIP LOD progression continues, at some level of detail texels have grown so large that the texture patch extends to the maximum visibility range required for the particular application. From here on, the remaining MIP levels can all be stored in a single final patch of memory. At run time, texture is managed so as to keep the eye point centered in a level-of-detail hierarchy of fully populated MIP texture. Total texture usage and overall system performance can be adapted to particular needs by varying the size of these patches. Determination of required texture pages is quickly done by reference simply to the location of the moving eye point.

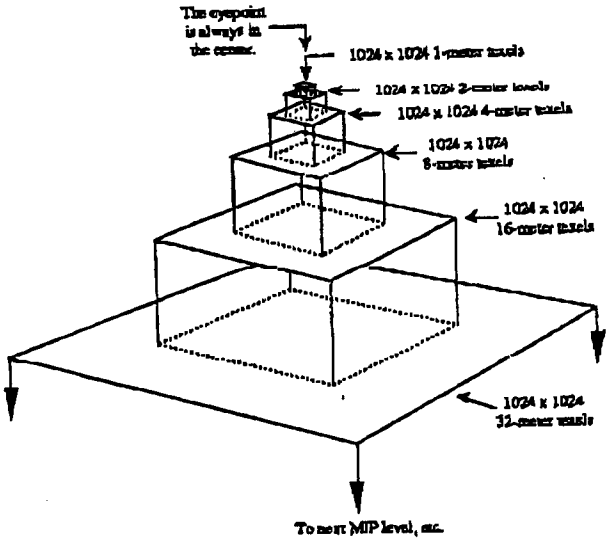

Fig. 2 Terrain texture MIP hierarchy.

Two nominal MIP patch sizes are currently offered: 512 and 1024 texels square. The eye point is always at the middle of the square, so texels at each MIP LOD are available either to 256 or 512 texels distant. An example using the 1024 patch size helps illustrate this characteristic of the system. Nominal texel size at the perimeter of the patch is about seven arc-minutes; however, at this range we would be almost completely transitioned to even coarser texels at the next MIP level (the texture LOD transition strategy is designed to ensure this). Since most display environments provide pixels that are considerably smaller than this, texture will look less sharp than it could (based strictly on aliasing considerations). However, most terrain is not viewed straight down, but out the windows of the aircraft, typically in a horizontal direction. Under these conditions, perspective compression of the terrain texture

often prevents us from using even the texels we have, and most of the displayed terrain texture is resolution-limited by aliasing considerations, not because it is outside the 512-texel range.

Storage And Paging Characteristics. Texture memory for each MIP LOD is pre-allocated by the system separate from the feature texture storage, guaranteeing adequate terrain texture storage space. Because the terrain texture is independent of the terrain polygonalization, texture paging is not affected by the organization or density of the terrain model. Texture paging load can be accurately determined from the data base design parameters (texel size, visibility range and eye point velocity). Run-time paging load will be linearly related to the eye point velocity. Terrain texture is paged independently for each MIP LOD, and this provides a large reduction in total paging load. Preferential paging of the lower LODs ensures a benign paging overload fallback--terrain would then be textured with a lower resolution version of the right texture.

Total hardware texture storage is dramatically reduced by managing each MIP LOD separately. If we need 1 meter texels out to a visibility range of 50 km, the following MIP structure will suffice: level 1 gets us out to 512 meters, level 2 to 1 km, level 3 to 2 km, etc. Level 8 gets us to 64 km, and all the remaining MIP LODs can be stored in level 9, for a total storage requirement of 9 megatexels. This is three orders of magnitude less than the 13 gigatexels cited earlier. Paging load is similarly reduced. At 185 meters/second, the amount of texture in level 1 that is changing is 185 X 1024 or about 200k texels per second. Half this amount is changing in level 2 since texels are twice as large; one fourth as many in level 3, etc., for a total paging load of 400k texels per second. This is typical of rates supported by the system, which has been optimized to support low altitude fast jet applications at usable texture resolutions. For applications that require texture compression to meet disk storage constraints, an option provides JPEG decompression in silicon. This further improves paging performance by reducing the quantity of data fetched from disk.

Terrain/Texture Independence. The terrain polygon model and the global terrain texture model can be developed independently and in parallel, with correlation enforced by reference to a common geodetic datum and transform. Their separate coordinate spaces are connected at a single data base node by specification of their relative origins and global scales. Datums of the two spaces are assumed to align, but scaling of the spaces can be independent. The relative origins of the terrain and texture models can be tweaked at run time without touching the data base. Thus the texture strategy is topologically and topographically independent of the terrain strategy. There is no impact on the choice of terrain skinning--regular or irregular approaches can be used as required. Local area updates are easy--changes to either strategy are independent and there are no unexpected ripple effects. It takes less time to develop or modify a data base that has global terrain texture. Development of the terrain texture model in a unified geodetic context helps enforce thematic correlation with the source data, and facilitates validation of the finished data base.

Simplified Texture Development. The system utilizes texture modeling tools developed and refined over the last several years, and previously reported on (Cosman, 1992). These tools work within an infinite continuous texture space, with file breakup and file handling hidden inside the tooling and the operating system. The modeler accesses data in a geodetic sense, by specifying the latitude and longitude of areas he wishes to develop or enhance. The specification of ground control points, registration of different image sources, and warp of imagery into the final reconciled form is not hindered by file partitioning or texture map boundary issues. The modeler can build the highest-LOD texture from a variety of irregular patches of source using schematic control to facilitate the assembly process. Since the vectorized feature data is in the same data space, reconciliation of source is easier and more intuitive. Streamlined batch processes that format the final IG-loadable model automatically create the MIP levels and (optionally) compress the texture. The development cost of global terrain texture is significantly reduced.

Better Visual Results. Like the single global terrain texture motif itself, the texture MIP LOD structure is complete and continuous from its highest resolution all the way to the single texel that represents the whole data base. The system supports an extremely wide dynamic range of viewing conditions, and can even accommodate orbit-to-landing scenarios.

The total absence of visual artifacts at either terrain polygon or texture map edges means the global texture motif is well behaved everywhere.

Source Material: The Final Frontier

An IG that can do global terrain texture efficiently is only half the solution. There are many technical and financial obstacles to the development of a photographically sourced high resolution large area texture motif. At the foot-to-meter texel sizes required, satellite data is too coarse (especially true-color data), while aerial photos are hard to get, expensive, and huge numbers of them are needed for such a data base. Given the practical limits of film emulsion technology, a typical aerial photo might cover less than 100 square miles of terrain at one meter resolution. Half a million square miles means 5,000 photographs, and each of these will come with its own individual set of technical and aesthetic warts which must be excised by human intervention.

A Photo Tells A Thousand Lies

Photographs are full of errors. They include limitations of the film technology itself (variations in color, saturation, intensity, contrast), problems in the camera system (hot spots, anamorphic distortions, chromatic aberration, focus variations), lack of cooperation by Mother Nature (illumination, clouds, seasonal variations, diurnal effects, weather, shadows, shading), imperfect subject material (perspective, layover, self-occulting problems, earth curvature distortions, and image motif lateral displacement due to terrain vertical relief), and fickle fate (lint and other blemishes that always seem to infect the stuff.)

Error correction generally requires a human in the loop--first to discern that something is wrong, and then to try to guide its correction. Sometimes this requires information that is unavailable or hard to get. Not every photo contains the small number of recognizable, map-correlatable, known-elevation control points that are essential for proper image registration. Camera models are never exact, and a terrain elevation model is essential if one wants to orthonormalize out image displacements due to terrain relief. The expertise of the operator highly influences the quality of the eventual product, and fixes always degrade the information content of the imagery while adding noise. It's very hard and very expensive to get to a single texture layer for the whole data base at acceptable quality and usably high resolution. Although a variety of tools exist to do portions of this, most are complex, expensive, and need to run on high-end workstation hardware to achieve acceptable throughput rates for such large areas.

The bottom line is simply that the IG is capable of presenting more and better texture than users can typically afford to create. Photographically specific texture is the right and essential thing to do for limited portions of the data base (typically take-off, landing, hover, nav points and targets) but becomes prohibitively expensive for whole data base use over large areas at consistently high resolutions.

The Themel® Alternative

High-end users of global terrain texture have been struggling with these issues for half a decade. One solution that was developed out of lessons learned in the SOFATS program was the theme-texture mechanism reported on in Cosman (1992). Themel® refers to a new texture construction and display method that assembles photo-derived texture from thematic (for example, DFAD) data at run time. An ESIG 3000 incarnation of this capability has been developed that uses optional dedicated hardware, and provides the same functionality as the ESIG 4000.

Theme texture is photo-realistic, even though it is not photo-specific. It is derived from and fully correlated with the primary data base thematic source--the vectorized feature data. Its creation is extremely rapid, fully automated, and there is a variety of existing world-wide data sources already available and usable. Source material costs can be nominal to near zero, and the resulting terrain texture motif is always consistent with the feature decoration source data.

Fig. 3 illustrates the Themel® process. Briefly, thematic source data such as DFAD is rasterized into a texture map whose texels denote the theme type or feature coding of each texel area. The resolution of the theme map is coarse by texture standards--typically 40 meters per texel in a fast-jet application--but appropriately accurate given the nature and quality of available thematic data. When theme

texels are paged into the IG, dedicated hardware is used to instantiate a block of photo-derived texture of the designated theme type in place of each single theme texel. The photo texture has much higher resolution than the theme texture--typically 1024 photo texels replace each theme texel. Special hardware algorithms are used to blend the boundaries between different theme types in ways that correlate to those theme types. Photo texel blocks are instantiated from much larger theme-specific motifs (up to 1024 by 1024 texels), so repetition is effectively suppressed. When combined with high-resolution "grit" texture, the resulting motif has an extremely wide dynamic range, and functions well from skid height to many miles away.

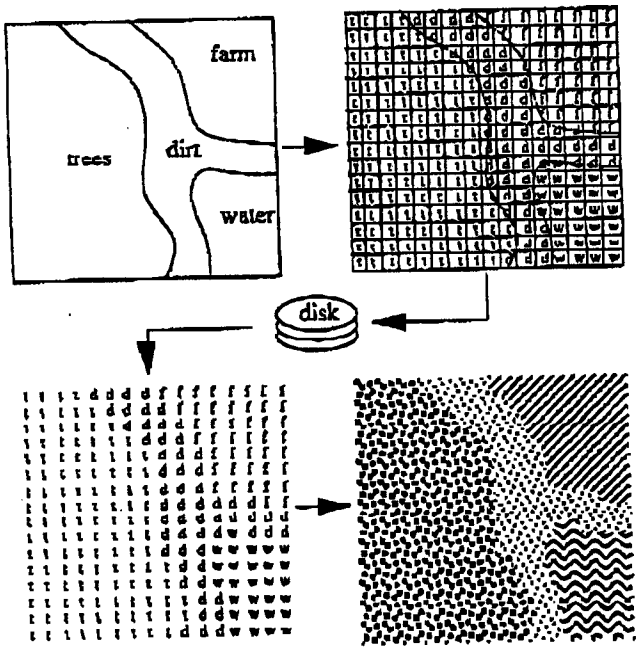

Fig. 3 Theme texture process.

Themels® provide huge reductions in the stored texture data--typically several thousand to one. The user has complete control over image quality and content issues because he constructs and tweaks only a small catalog of photo-derived texture theme patches. He can optimize the color, contrast, saturation and general visual behavior of each theme type, and will typically get better and more consistent results (in an aesthetic sense) than with photographic source materials. He can also provide alternate run-time libraries of the photo-texel replacement motifs to allow wholesale modification of the data base without any reprocessing. This can be used to select diurnal or seasonal effects. The tremendous cost savings of Themels® allows the data base developer to spend more time on the areas that do require photographically specific texture.

Conclusions

Compared to patchwork texturing mechanisms, dedicated global terrain texture hardware provides important visual and computational benefits. Visual advantages result from treating the global terrain texture as a single motif whose important coherence properties are respected and preserved. Significant run-time performance benefits result from exploiting level-of-detail coherence to greatly reduce paging rates, total IG texel storage, and the difficulty of determining what should be in the hardware memories. Dedicated global terrain texture hardware can also be more cost-effective when total life-cycle system costs are considered, since modest increases in hardware cost are quickly offset by savings in the creation and support of data base models. Important other benefits include more agility in keeping simulated environments current, and greater flexibility to employ limited resources in ways that maximize return to the user, such as the Themel® option to reduce source material costs.

References

Cosman, M., Mathisen, A. and Robinson, J. (1990). A New Visual System To Support Advanced Requirements. Proceedings of the 1990 IMAGE V Conference (pp. 370-380). Tempe, Arizona: The IMAGE Society.

Cosman, M. (1992), Mission Rehearsal Modeling. Proceedings of the 1992 IMAGE VI Conference (pp. 413-425). Tempe, Arizona: The IMAGE Society.

Norton, A., Rockwood, A., and Skolmoski, P. (1982). Clamping: A Method of Antialising Textured Surfaces by Bandwidth Limiting in Object Space. Computer Graphics, July 1982.

Williams, L. (1983). Pyramidal Parametrics. Computer Graphics, July 1983.